Efficiently allocating distributed caching resources in future smart networks

Hamza Ben Ammar, Yassine Hadjadj-Aoul, Soraya Aït-Chellouche

Efficiently allocating distributed caching resources in future smart networks

Hamza Ben-Ammar, Yassine Hadjadj-Aoul and Soraya Ait-Chellouche
Univ Rennes, Inria, CNRS, IRISA
Emails: {hamza.ben-ammar, yassine.hadjadj-aoul, soraya.ait-chellouche}@irisa.fr

***Abstract*—During the last decade, Internet Service Providers (ISPs) infrastructure has undergone a major metamorphosis driven by new networking paradigms, namely: Software Defined Networks (SDN) and Network Function Virtualization (NFV). The upcoming advent of 5G will certainly represent an important achievement of this evolution. In this context, static (planning) or dynamic (on-demand) caching resources placement remains an open issue. In this paper, we propose a new technique to achieve the best trade-off between the centralization of resources and their distribution, through an efficient placement of caching resources. To do so, we model the cache resources allocation problem as a multi-objective optimization problem, which is solved using Greedy Randomized Adaptive Search Procedures (GRASP). The obtained results confirm the quality of the outcomes compared to an exhaustive search method and show how a cache allocation solution depends on the network's parameters and on the performance metrics that we want to optimize.**

***Index Terms*—Multi-cache systems, Cache allocation, Multi-objective optimization, GRASP, VNF placement.**

I. Introduction

The endless race of content providers towards even more content and more quality is increasing data consumption as never before. This growth is such that it is undermining the infrastructure of Internet Service Providers (ISPs). The latter organize themselves by developing caching capabilities, increasingly close to end-users in order to relieve the load on their networks [1]. These recent developments allow them to share their infrastructure by offering these caching capabilities to third parties, which makes it possible to multiply sources of revenues. The virtualization of network functions (NFV), especially caching, as well as the dynamic orchestration of resources bring much more flexibility to the infrastructure and their management. These features will be, therefore, major drivers that will certainly accelerate these developments and make them available to content providers [2]. These building blocks represent, indeed, the main foundation of the future core of the mobile network, namely 5G.

Setting up an optimal caching system within a network infrastructure, up to the edge of the latter at the Multi-access Edge Computing (MEC) level, is very complex and remains an open issue in the literature [3]. In fact, if we want to optimize the use of caches, we will tend to put everything in the same place at the exit of the network (at Point of Presence (PoP) level), where users requests will be aggregated. This allows to maximize the hit rate, but has the drawback of overloading the network. However, if one wants to maximize user satisfaction while at the same time reducing network load, the ideal would be to put everything on the edge. The problem here is that there will be more redundancy at the edge because of the multitude of access networks and the origin server will be more solicited.

We investigate, in this paper, the optimal placement of caching resources. The proposed approach solves the trade-off between minimizing the hit rate in the origin server (with the risk of degrading the quality of experience) and minimizing the distance between clients and the requested contents (with the risk of ineffective caching). For this purpose, we propose a model considering, at the same time, the different objectives mentioned below, namely: (i) latency (or distance), in order to bring the content closer to the users, and (ii) the cache hit in the ISP, in order to request the origin server the least. We model the cache allocation problem as a multi-objective integer nonlinear program. Since it is an NP-hard problem, we propose the use of the Greedy Randomized Adaptive Search Procedures (GRASP) [4], which has shown its effectiveness in solving combinatorial optimization problems.

The reminder of this paper is organized as follows. In section II, we present the related work. Section III provides the detailed description of the cache placement problem and the proposed solution. Then, the evaluation of our proposal is displayed in section IV. Finally, section V summarizes the achievements in this paper and introduces our future work.

II. Related Work

Many studies have investigated the problem of cache resource allocation and placement in the context of multi-cache networks (e.g., CDN, ICN, etc.). Krishnan et al. in [5] studied the cache location problem in the case of transparent in-route caches in the context of web caching. The problem was modeled as a k-median problem, where the objective is to minimize the network traffic flow, and solutions were proposed based on dynamic programming and greedy heuristics. The study in [6] was probably the first attempt to investigate the cache allocation problem in CCN. They used in their study different metrics to measure the centrality of routers like degree, closeness and betweeness in order to decide where the cache should be distributed along the network's nodes. They suggest that deploying more cache resources at the core nodes of the network is better than at the edge. In later works [7] [8], the authors have concluded the opposite, suggesting that placing larger caches at the edge is more effective. Wang et al., in their work [9], have studied the impact of content popularity distribution on caching performance in CCN. They

show that placing caches into the network core is better suited for content requests with uniform distribution and that in case of highly skewed popularity demands patterns, pushing cache resources to the edge yields better performance.

Considering a single metric clearly reduces the complexity of the problem, but has led many studies to find contradictory results. There are many aspects in our proposal that makes it different from the existing works. We propose a versatile tool that takes into account more than one performance metric to solve the cache allocation problem and it can be tuned in order to seek some specific results. In addition, the solutions are built by measuring the contribution of all the nodes by the means of an analytic model capable of estimating the network performance, which will allow to take into account the impact of a node's performance on the others.

III. Multi-objective cache placement strategy

A. Problem description

In previous works that addressed the cache allocation problem, only one performance metric is generally considered to generate a solution. In this work, we use jointly the following two performance metrics to evaluate the cache gain: content provider load and average distance ratio. The first metric represents the amount of contents that were served by the origin server over all the requests sent in the network. The second one depicts the average distance travelled to get contents in the network over the obtained distance without caching. We chose these two metrics for their importance in representing the cost and gain obtained from in-network caching. A high content provider load means most of the requested contents are not served by the intermediate caches. Accessing the main source of contents is expensive for network operators, this is why it is important for them to keep the content provider load as low as possible. On the other hand, a low average distance to get contents means a better Quality of Experience for users. Hence, a good cache allocation strategy should find the best trade-off between these two metrics.

B. System assumptions

Let $G = (V, E)$ be the graph representing a network of caches, where $V = \{v_1, ..., v_M\}$ depicts the nodes of the network and $E \subset V \times V$ is the set of links connecting the nodes. Each node is equipped with a caching module used to store contents locally. Let $C = \{c_1, ..., c_R\}$ be the set of the catalog's contents available for the users. We assume that all the accessible contents in the system have an identical size and are divided into small packets or chunks, which are in turn of the same size. The cache capacity is then expressed in terms of the number of contents or chunks that can be stored. All the available contents are stored permanently at one or more servers attached to some nodes within the network.

Clients, which are attached to the network nodes, send requests into the network looking for contents. The pattern of these requests is characterized by the Independent Reference Model (IRM) [10]. Suiting the IRM model, users generate an independent and identically distributed sequence of requests from the catalog C of R objects. Specifically, the probability p_r to request an item c_r from the set of available contents in the network is constant and follows a popularity law, where the contents are ranked decreasingly according to their popularity from one to R. As already argued in many previous studies, the latter fits the Zipf law [10]: the probability to request the content of rank r is: $p_r = r^{-\alpha} / \sum_{i=1}^{R} i^{-\alpha}$, where α, the skew of the distribution, depends on the type of the accessible objects. In the present work, the LRU algorithm is used to manage the node's content store and the Leave Copy Everywhere (LCE) scheme is employed as a caching policy. LCE is the default caching scheme used in multi-cache networks, where the objects received by each node is always cached. Note that the proposed solution may work similarly with other caching strategies.

C. Problem formulation

A cache allocation solution (typically taken by an NFV orchestrator), can be defined by a vector $X = (x_1, \ldots, x_M)$, where x_i represents the amount of storage capacity placed in node v_i. To compute the content provider load or the average distance ratio of a configuration of caches X, we use our model MACS (Markov chain-based Approximation of Caching Systems), which we presented in previous works [11] [12]. MACS is an analytic tool that allows us to estimate the cache hit ratio of an interconnection of caches, which can be used to compute other performance metrics like the content provider load and the average distance ratio. By using MACS and for each cache allocation configuration X, we can evaluate the performance of the whole system in its steady-state and not just during a transient phase.

Since in this work the caching capacity of a Virtual Network Function (VNF) or a physical cache is expressed in terms of the number of contents that can be stored, then we have $x_i \in \mathbb{N}$. As we measure the caching gain through evaluating the content provider load and the average distance ratio in the network, our objective is to find a cache distribution solution such that the evaluation metrics that we have chosen are optimized (i.e. minimized). The cache placement is then formulated as a multi-objective optimization problem as follows:

$$\begin{aligned} &\textit{minimize} && f_1(X), f_2(X) \\ &\textit{subject to} && \sum_i x_i \leq T_c, i = 1, \ldots, M, \\ & && x_i \geq 0, i = 1, \ldots, M, x_i \in \mathbb{N}. \end{aligned} \tag{1}$$

The value of T_c represents the total cache resources available in the network. The expressions $f_1(X)$ and $f_2(X)$ are expressed as a percentage and represent, respectively, the content provider load and the average distance ratio of a cache placement configuration X. Using MACS, we can calculate the cache hit and cache miss ratios of the network's nodes. Then, we can compute $f_1(X)$ and $f_2(X)$ as follows:

$$\begin{aligned} f_1(X) &= \Big(\prod_{i=1}^{s} P_{miss}(i)\Big) \times 100, \\ f_2(X) &= \frac{\sum_{i=1}^{s} \Big(\big(\prod_{j=1}^{i-1} P_{miss}(j)\big) \times P_{hit}(i) \times Dist(i)\Big)}{Dist(s)} \times 100. \end{aligned} \tag{2}$$

The values $P_{miss}(i)$ and $P_{hit}(i)$ represent, respectively, the cache hit and cache miss of a network's node v_i. The expression *Dist*(i) is the distance from where the clients requests were generated to the node v_i. The index s represents the nearest node v_s to the clients to which a content provider is attached (i.e. where a permanent copy of the contents is available). In the definition of $f_1(X)$ and $f_2(X)$ and for sake of clarity, we supposed that the network is formed by a line of s nodes numbered from i to s where the clients are attached to node v_1 and the content repository is located just after node v_s. It has to be noted that of course, we can compute these metrics or other ones in any type of network topology.

Since here we are dealing with a multi-objective optimization problem, in which we want to minimize $f_1(X)$ and $f_2(X)$, the solutions will be a set of *efficient* points usually called the Pareto frontier. Pareto efficiency or optimality implies that a solution to a multi-objective problem is such that no single objective can be improved without deteriorating another one. In our case, a solution X^* is said to be efficient if there is no other solution X such that $f_1(X) < f_1(X^*)$ and $f_2(X) \leq f_2(X^*)$ or $f_2(X) < f_2(X^*)$ and $f_1(X) \leq f_1(X^*)$. Given that integer nonlinear programming is an NP-hard problem, solving the cache allocation problem as we modeled below will come at a very high computational cost. Therefore, we propose the use of the meta-heuristic GRASP to solve it.

D. Solving cache allocation problem using GRASP

The Greedy Randomized Adaptive Search Procedures or GRASP [4] is an iterative process, where each iteration consists basically of two steps: construction and local search. The construction phase seeks to build a feasible solution using a greedy randomized approach, whose neighborhood will be investigated during the local search in order to find a local optimum. The best overall solution is, then, kept as the final result. During the construction phase, the candidate set *CS* will be the list of cache placement configurations X_i where a partial resource that we denote P_c, is taken from the total available cache T_c and allocated to one of the network's nodes. If we have for example a network with three nodes, $T_c = 100$ and $P_c = 10$, the initial candidate set will then be: $CS = \{(10, 0, 0), (0, 10, 0), (0, 0, 10)\}$.

Each candidate is then evaluated with a greedy function in order to build a restricted candidate list *RCL*, which will contain some of the candidate set who have the best evaluation values (e.g., $RCL = \{(10, 0, 0), (0, 10, 0)\}$). The limitation criteria of the list cardinality can be either based on the number of elements or based on their quality. The elements added to the *RCL* list will then be those having an evaluation value inferior to the threshold (i.e., $f(X) \in [f^{\min}, f^{\min} + \beta(f^{\max} - f^{\min})]$). The value of β will control the insertion condition of candidate elements to *RCL*. The case $\beta = 1$ is equivalent to a random construction, while $\beta = 0$ corresponds to a pure greedy algorithm. Once *RCL* is built, an element from it is randomly selected and added to the partial solution. The candidate list *CS* and the evaluation function are then updated and the construction is repeated until the total use of the cache budget T_c. If we consider for example that the second element from *RCL* has been chosen, the current partial solution will then be $S = (0, 10, 0)$ and the new candidate list will contain: $CS = \{(10, 10, 0), (0, 20, 0), (0, 10, 10)\}$.

Once the cache budget T_c is distributed, the local search will seek to improve the generated solution (e.g., $S = (20, 40, 40)$) by evaluating its neighborhood. We used for the neighborhood search the following procedure: starting from the solution generated by the greedy randomized construction, we transfer an amount of cache P_c from one node to another looking for a configuration whose cost function value is lower than the current one. If a better neighbor is found, we keep it and repeat the search in order to find a better solution (e.g., $Neighbor(S) = (10, 50, 40), (30, 30, 40), etc.$). The best overall solution *S* is then returned as the output of GRASP.

When dealing with multi-objective GRASP [13], the candidate elements will be evaluated with more than one function (in our case, we have $f_1(X)$ and $f_2(X)$) and there are many methods that can be used to compute the outcome of each configuration in the construction and local search phases (see [13] for more details). For example, and in our case, one can use $f_1(X)$ in the construction phase and $f_2(X)$ during the local search and vice versa, or choose randomly between the objective functions during each GRASP iteration. We can also consider a weighted combined method ($f(X) = w_1 f_1(X) + w_2 f_2(X)$). The choice of which method to be used to evaluate a candidate solution can depend on some desired results that should be achieved or some constraints that should be respected. In our work, we considered many methods to generate the solutions based on the defined objective functions $f_1(X)$ and $f_2(X)$ but due to the lack of space, only the results of one approach will be presented. It consists on using in each iteration one objective function ($f_1(X)$ or $f_2(X)$) during the construction phase. The other one is then used during the local search. Then, we alternate between the selected functions for each phase in the next iteration. This approach will allow us to produce diversified solutions of good quality.

IV. Performance evaluation

A. Model configuration

The different tests were conducted on a typical three-level network that contains 21 nodes forming a perfect 4-ary tree topology where the distance and the latency between each two adjacent nodes are the same. We considered in our experiments a catalog of contents containing $20,000$ 1-chunk sized objects whose popularity distribution follows the Zipf's law. Permanent copies of the available contents are hosted on one repository attached to the root node of the network and the users are attached to the network's leaves (i.e. level 1 of the network). We tested different values of the Zipf law's skew parameter α going from 0.8 to 1.2. The parameter β that controls the amounts of greediness and randomness in GRASP was set to 0.5 and the total cache T_c is expressed as a ratio of the catalog size. For sake of simplicity, the cache resources are allocated in a way that the nodes located on the same level of the network have the same cache size. Since there are 16

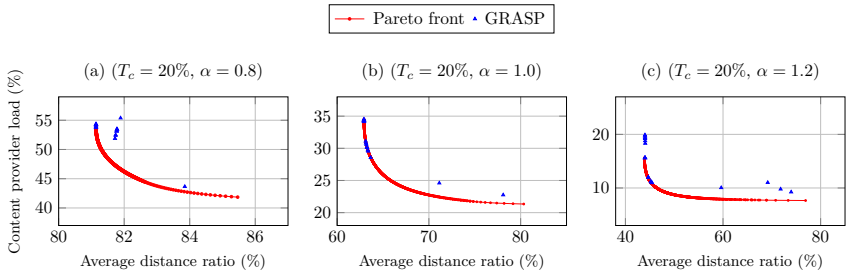

Figure 1: Comparison between the Pareto front and GRASP solutions

nodes on the first level of the network, P_c should be a multiple of 16 and in our case, it was set to 16. Thus, a cache resource placement can be described as $X = (x_1, x_2, x_3)$, where x_i represents the total cache allocated at level i of the network.

B. Model results and analysis

In figure 1, we compare the network's performance metrics of the different cache distribution solutions generated by GRASP (20 iterations) with the Pareto front. We can see from the plots how the solutions produced by GRASP are very close to the set of dominant points in the different tested scenarios, which reflects the good quality of the metaheuristic outcomes.

In Table I, we display the cache allocation solutions generated by GRASP in different configurations (only the results of three iterations of GRASP are shown) using a separate evaluation functions approach. Depending on the Zipf law's parameter α and the total cache resources available T_c, different solutions will be generated by GRASP. The choice of the final solutions between the different proposed ones can be based on some constraints that should be respected. For example, if $\alpha = 1.2$ and $T_c = 20\%$ and if the priority is to minimize the content provider load ($f_1(X)$), then the allocation $X = (1952, 1040, 1104)$ is the best one.

We can see from the exposed different results that there is no absolute solution for the cache allocation problem and it is not a question about whether to cache at the edge or in the core. Depending on the network's configuration and the objectives that one wants to achieve, multiple solutions can be adopted. Other metrics can be used as evaluation functions during the building of the solutions instead of those that we used in this work like for example the financial cost of cache resources deployment. The aim of our proposal is to give a tool capable of efficiently allocating distributed caching resources and versatile enough to adapt to specific desired network performance and constraints.

Table I: Cache allocation solutions using GRASP with a separate evaluation functions approach ($T_c = 20\%$)

α	X	$f_1(X)$	$f_2(X)$
0.8	(320,1696,2080)	53.47%	81.78%
	(768,1376,1952)	54.80%	81.15%
	(736,1280,2080)	53.84%	81.14%
1.0	(1536,1168,1392)	34.39%	62.92%
	(944,880,2272)	28.57%	63.65%
	(1136,864,2096)	29.54%	63.38%
1.2	(1952,1040,1104)	15.62%	44.00%
	(1760,2176,160)	19.88%	44.00%
	(1456,720,1920)	12.02%	44.67%

V. Conclusion

We address in this paper the problem of allocating efficiently cache resources in multi-cache networks. We first model it as a multi-objective optimization problem where our analytic tool MACS is used to evaluate the objective functions. Our formulation of the problem turned out to be an NP-hard problem and thus, we propose an adaptation of the GRASP metaheuristic to solve it. The experimental results have showed how the outcomes of our algorithm are very close to the optimal ones. The versatility of our proposal allows it to be applied to any network topology and it can be used with metrics different from the ones used in this paper. We intend in the future to conduct more experiments on the cache allocation problem using our proposal by considering more network performance metrics and testing other caching schemes.

References

[1] P. Bertin, T. Mamouni, and S. Gosselin, *Next-Generation PoP with Functional Convergence Redistributions*, Jan. 2017, pp. 319–336.

[2] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Communications Surveys Tutorials*, Apr. 2018.

[3] K. Reddy, S. Moharir, and N. Karamchandani, "Effects of storage heterogeneity in distributed cache systems," in *WiOpt 2018*.

[4] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Opt.*, vol. 6, pp. 109–133, Mar. 1995.

[5] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Trans. Netw.*, pp. 568–582, Oct. 2000.

[6] D. Rossi and G. Rossini, "On sizing ccn content stores by exploiting topological information," in *INFOCOM Wksh.*, Mar. 2012, pp. 280–285.

[7] I. Psaras, W. K. Chai, and G. Pavlou, "In-network cache management and resource allocation for information-centric networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 25, pp. 2920–2931, Nov. 2014.

[8] S. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable icn," *SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 147–158, Aug. 2013.

[9] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and evaluation of the optimal cache allocation for content-centric networking," *IEEE Transactions on Computers*, vol. 65, pp. 95–107, Jan. 2016.

[10] C. Fricker, P. Robert, J. Roberts, and N. Sbihi, "Impact of traffic mix on caching performance in a content-centric network," in *INFOCOM Wksh.*, Mar. 2012, pp. 310–315.

[11] H. Ben-Ammar, S. Ait-Chellouche, and Y. Hadjadj-Aoul, "A Markov chain-based Approximation of CCN caching Systems," in *IEEE Symposium on Computers and Communications*, Jul. 2017, pp. 327–332.

[12] H. Ben-Ammar, Y. Hadjadj-Aoul, G. Rubino, and S. Ait-Chellouche, "A versatile markov chain model for the performance analysis of CCN caching systems," in *IEEE GLOBECOM 2018 (To appear)*.

[13] R. Martí, V. Campos, M. Resende, and A. Duarte, "Multiobjective grasp with path relinking," *Eu. Jour. of Op. Res.*, vol. 240, pp. 54–71, Jul. 2014.

We would like to thank the Lannion-Trégor Community and the Regional Council of Brittany for their financial support.